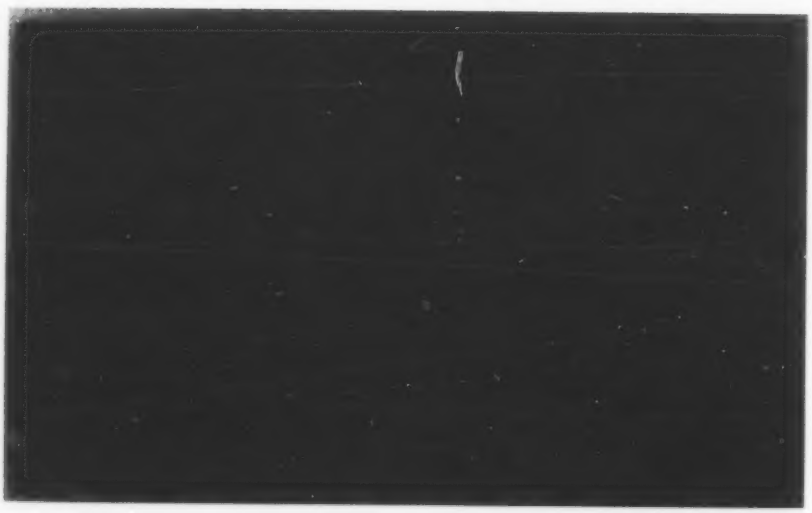
CENTRE DE RECHERCHE SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE



CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

93-01752/

CRT-537

An Active Constraints Newton Algorithm for The Spatial Price Equilibrium Problem

by

Omar Drissi-Kaïtouni
Michael Florian

This research was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

Centre de recherche sur les transports — Publication #537
Département d'informatique et de recherche opérationnelle — Publication #628

July 1987
Revised July 1988 and December 1991

Dépôt légal – 2e trimestre 1992

ABSTRACT

We consider the general spatial price equilibrium problem. In order to solve such problems, we develop a Newton type algorithm that is combined with an active constraints strategy to handle the nonnegativety constraints or upper bounds on the variables. At each iteration of the algorithm, the choice of the initial active set is based on the value of the gradient and the dual variables. Two variants of the algorithm are presented: one in which the Newton equations are solved by the Gauss method and one in which these equations are solved by the conjugate gradient method. Computational results demonstrate the efficiency of the method.

Key words: Spatial Price Equilibrium on a General Network, Network Equilibrium, Newton, Active Constraints, Conjugate Gradient.

RÉSUMÉ

Nous considérons le problème d'équilibre spatial général. Pour résoudre ce type de problème , nous avons développé un algorithme de type Newton combiné à une stratégie de contraintes actives pour tenir compte des contraintes de non-négativité et de capacité. A chaque itération de l'algorithme, le choix de l'ensemble initial de contraintes actives est basé sur la valeur du gradient et des variables duales. Deux variantes de l'algorithme sont présentées: une où les équations de Newton sont résolues à l'aide de la méthode de Gauss, et une autre où ces équations sont résolues à l'aide de la méthode du gradient conjugué. Les résultats démontrent l'efficacité de la méthode.

Mots-clés: équilibre spatial sur réseau général, équilibre de réseau, Newton, contraintes actives, gradient conjugué.

INTRODUCTION

We present in this paper a "Newton-Active Constraints" algorithm for the solution of the spatial price equilibrium problem formulated in the space of arc flows (see Friesz and Tobin (1983)). As it is well known, the descriptive equilibrium state of quantities bought, sold and transported on the links of a network are the solution of the problem

$$\text{Min } F(d,v,b) = \sum_l \int_0^{d_l} S_l(x)dx + \sum_{a\in A} \int_0^{v_a} C_a(y)dy + \sum_l \int_0^{b_l} I_l(z)dz \quad (1)$$

$$\text{subject to} \quad b_l + \sum_{a\in A_l} v_a - \sum_{a\in B_l} v_a - d_l = 0, \quad \forall l \quad (2)$$

$$\text{and} \quad d_l \geq 0, \; b_l \geq 0 \;\; \forall l; \quad v_a \geq 0, \quad \forall a. \quad (3)$$

$S_l(.)$ is the supply function at origin l, $C_a(.)$ is the average cost function for transportation on arc a and $I_l(.)$ is the inverse demand function at destination l; d_l is the quantity supplied at origin l and b_l is the quantity bought at destination l; v_a is the transportation flow on arc a. In order to state the conservation of flow equations, B_l is the set of arcs which have node l as tail node and A_l is the set of arcs which have node l as head node.

This version of the problem is usually referred to as the General Spatial Price Equilibrium Problem (GSPEP) since it is formulated on a general network. Friesz and Tobin (1983) proposed the solution of this problem by the use of the reduced gradient method implemented by Murtagh and Saunders (1977) in the MINOS code.

The motivation for carring out the adaptation of a Newton type method for this problem is to obtain a more efficient algorithm which exploits the information available from the Hessian of the objective function (1) and hence improve the computational results that may be obtained by using MINOS.

The paper is organized as follows. In the next section, we consider a general convex cost minimization problem of the form

$$\min \ f(x) \tag{4}$$

$$\text{subject to } Ax = b \tag{5}$$

$$\text{and } x \geq 0 \tag{6}$$

(where $x \in R^n$, $A : m \times n$, $m \leq n$ is a full rank matrix, $b \in R^m$ and $f : \Phi \rightarrow R$ is strongly convex and twice differentiable on $\Phi = \{x \in R^n,\ Ax = b,\ x \geq 0\}$). Then, after a short review of Newton's method for problems with equality constraints, we consider the solution of the resulting quadratic subproblem by using a Newton Active Constraints method, such as described in general by Fletcher (1972) and Murray and Wright (1982), for its solution. Our approach uses a particular choice of the set of initial active constraints to initiate the computations. Then, we consider a variant of the algorithm where the resulting quadratic subproblems are not solved to ϵ-optimality. Then we adapt the algorithm for the General Spatial Price Equilibrium Problem and present computational results which demonstrate that both the algorithm and its variant are much more efficient than the reduced gradient method, as implemented in MINOS, for the solution of this problem. We conclude the paper by pointing out other nonlinear cost network optimization problems where this type of algorithm may be very effective.

1. A Review of Newton's Method for Constrained Convex Cost Optimization

We review briefly the solution of (4)–(6) by Newton's method. The general algorithm may be stated as follows:

Step 0 $k = 1$, determine a feasible solution x^1

Step 1 Solve the quadratic approximation Q_k at x^k.

$$\underset{y}{\text{Min}}\ \nabla f(x^k)^{\mathrm{T}}(y-x^k)+\frac{1}{2}(y-x^k)^{\mathrm{T}}\nabla^2 f(x^k)(y-x^k) \tag{7}$$

$$\text{subject to} \quad Ay=b \tag{8}$$

$$y\geq 0 \tag{9}$$

Step 2 $d^k=y-x^k$; if $|\nabla f(x^k)^{\mathrm{T}}d^k|\leq\epsilon$ STOP;

otherwise continue.

Step 3 Find $\alpha^k=\underset{\alpha}{\text{argmin}}\, f(x^k+\alpha d^k)$ subject to $x^k+\alpha d^k\geq 0$.

Step 4 $x^{k+1}=x^k+\alpha^k d^k$, $k=k+1$, and return to step 1.

The quadratic programming subproblem of step 1 is solved by a Newton Active Constraints (NAC) algorithm, which is a particular adaptation of the Newton method.

2. Newton's Method for Problems with Equality Constraints

We consider in this section of the paper the solution of (4)–(5) by Newton's method. This consists in solving a sequence of quadratic cost optimization problems. The general iteration of this algorithm, denoted by index k, consists then in solving the problem

$$\text{Min}\ \ \nabla f(x^k)^{\mathrm{T}}y+\frac{1}{2}(y-x^k)^{\mathrm{T}}\nabla^2 f(x^k)(y-x^k) \tag{10}$$

$$\text{subject to} \quad Ay=b \tag{11}$$

where $\nabla f(x^k)$ is the gradient of f evaluated at the current feasible solution x^k (which we will also denote as ∇f_k) and $\nabla^2 f(x^k)$ is the Hessian matrix of f evaluated at x^k (which we will also denote as D_k).

The Kuhn-Tucker optimality conditions for the problem (10)–(11) are, for the solution (y^k, λ^k)

$$\nabla f_k + D_k(y^k - x^k) + A^{\mathrm{T}}\lambda^k = 0 \tag{12}$$

$$Ay^k = b \tag{13}$$

Let $d^k = (y^k - x^k)$ be the Newton direction of descent. (12) and (13) may be rewritten as

$$D_k d^k + A^{\mathrm{T}}\lambda^k = -\nabla f_k \tag{14}$$

$$Ad^k = b - Ax^k \tag{15}$$

We recall (see Luenberger (1984), p. 424) that if $A : m \times n$, $m \leq n$ is a full rank matrix and $D : n \times n$ is a positive definite matrix, then

$$H = \begin{pmatrix} D_k & A^{\mathrm{T}} \\ A & 0 \end{pmatrix}$$

is a nonsingular matrix and, therefore, system (14)–(15) has a unique solution. It follows that if $f \in C^2$ is a strongly convex function of $x \in E^n$ and $A : m \times n$, $m \leq n$ is a full rank matrix, then $y^k = x^k + d^k$ is the optimal solution of (10)–(11), where λ^k is the vector of optimal multipliers associated to (11). Furthermore, let $x^k \in R^n$ and consider the Newton iteration (12)–(13); then y^k is feasible. If $d^k \neq 0$,then d^k is a feasible direction of descent for (10)–(11) and if $d^k = 0$, then x^{k+1} is its optimal solution and λ^k, the vector of optimal multipliers.

We note that for quadratic (and convex cost) optimization problems that are subject only to equality constraints, Newton's algorithm finds a feasible solution in the first iteration (starting from an arbitrary x^1, $x^2 = y^1$ is feasible); for all subsequent iterations the Newton direction is a feasible direction of descent. Clearly, the adaptation of the method for inequality constraints would seek to retain, as much as possible, these "good" properties.

3. A Newton-Active Constraints Approach for Quadratic Programs with Nonnegativity Constraints

We reconsider in this section the problem (4)–(5) with nonnegativity constraints $y \geq 0$ (9). These nonnegativity constraints may "block" the Newton direction d^k for (10)–(11), when one or more of these constraints become binding or active. d^k, as computed in (14)–(15) ensures that (11) are satisfied, however the nonnegativity constraints may be violated by $x^{k+1} = x^k + \alpha^k d^k$ for a step size α^k, $\alpha^k > 0$ unless an appropriate bound $\bar{\alpha}^k$, $\alpha^k \leq \bar{\alpha}^k$, is imposed on the step size. d^k is feasible for all $\bar{\alpha}^k > 0$ if $d^k = 0$, however if $d^k \neq 0$ then $x^k + \bar{\alpha}^k d^k$ remains feasible if, for all variable indices j

$$\bar{\alpha}^k \leq \min_j \left(-\frac{x_j^k}{d_j^k} \right) \quad \text{for } d_j^k < 0. \tag{16}$$

We shall refer to a variable x_j, $x_j^k = 0$ and $d_j^k < 0$ as a "blocking" variable for the Newton direction d^k. Let D_k be, as before, the Hessian matrix, which is diagonal for (7), with elements d_j^k, that is $D_k = \text{diag}\{ D_j^k \}$ and $D_j^k > 0$ since f is strongly convex. Then

$$d_j^k = -\frac{1}{D_j^k} \left(A_j^{\mathrm{T}} \lambda^k + \nabla f_j(x^k) \right), \tag{17}$$

where A_j is the j th column of A. If $d_j^k < 0$ then $\left(A_j^{\mathrm{T}} \lambda^k + \nabla f_j(x^k) \right) > 0$.

The Active Constraints algorithm that we develop in this section is based on the following observation. If the solution at iteration k is in the neighbourhood of the optimal solution of (7), it is reasonable to suppose that λ^{k-1} is a good approximation of λ^k. If so, then the value of $\left(A_j^{\mathrm{T}} \lambda^{k-1} + \nabla f_j(x^k) \right)$ is a good approximation of $\left(A_j^{\mathrm{T}} \lambda^k + \nabla f_j(x^k) \right)$ and it is likely that d_j^k and $\left(A_j^{\mathrm{T}} \lambda^{k-1} + \nabla f_j(x^k) \right)$ are of opposite sign. Based on this argument, let I and J be two sets of indices defined as follows:

$$\left\{ \begin{array}{l} I = \{ j \in N = \{ 1, 2, \ldots, n \} \mid x_j = 0,\ g_j^k = \left(A_j^{\mathrm{T}} \lambda^{k-1} + \nabla f_j(x^k) \right) > 0 \} \\ J = N - I \end{array} \right\} \tag{18}$$

where $j = 1, 2, \ldots, n$ are the indices of the variables and N is the set of variables of cardinality $|N|$.

The variables that correspond to $j \in I$ are likely to be those that "block" the Newton direction; those that correspond to $j \in J$ are likely to be those that are likely to yield a feasible direction of descent, that is either $x_j^k = 0$ and $d_j^k \geq 0$ or $x_j > 0$.

The algorithm that we suggest is based on the restriction of the Newton direction to the variables j, $j \in J$. At each iteration the direction p is defined to be

$$p = \begin{cases} p_j = 0 & j \in I \\ p_j & j \in J \end{cases} \tag{19}$$

where p_j, $j \in J$, is the Newton direction obtained by solving the system (14)–(15) restricted to the variables in the set J, for which the nonnegativity constraints are relaxed.

Our approach is different than other approaches (see Fletcher 1972, p. 289 and Murray and Wright, 1982, pp. 69–70) in the way that the initial set of active constraints is identified; rather than identifying one constraint at a time, an initial set is identified.

In general, this initial choice of the sets I, J does not insure that the direction p_J is always feasible, since λ^{k-1} may not be a good approximation of λ^k, particularly when x^{k-1} is not in a neighbourhood of the optimal solution. Thus, if "blocking" occurs, the sets I, J are modified as follows:

$$\left\{ \begin{array}{l} I \leftarrow I \cup \{ j \mid x_j = 0,\ p_j < 0 \} \\ J \leftarrow N - I \end{array} \right\} \tag{20}$$

and the Newton direction is recomputed as in (16). By adapting this strategy, it is possible nevertheless to augment the set I by a sufficient number of variables in

order to render the system

$$A_I p_I = 0$$

$$p_I = 0$$

singular, where A_I are the columns of A that correspond to variables $j,\ j \in I$ (and A_J are the columns of A that correspond to variables $j,\ j \in J$). If this occurs, one or more redundant constraints $p_q = 0,\ q \in I$ must be removed, which is equivalent to removing one constraint at a time from the active set I.

The solution algorithm for problem (7)–(9), which we denote NAC, is stated next.

Step 1-0 (Initialization)

$y^0 = x^k, \quad l = 1$

Define I and J as follows:

$I = \{ j \in N \mid x_j^k = 0,\ g_j^k = (A_j^{\mathrm{T}} \lambda^{k-1} + \nabla f_j^k(x^k)) > 0 \}$

$J = N - I$

Step 1-1 (Compute descent direction)

Find p^l as the solution of

Min $\frac{1}{2} p^{l^{\mathrm{T}}} D_k p^l + \nabla f_k^{\mathrm{T}} p^l$

subject to $Ap^l = 0; \quad p_l^l = 0.$

If $p^l = 0$ go to step 3.

Step 1-2 (Compute optimal step size)

$y^{l+1} = y^l + \alpha^l p^l, \quad \alpha^l = \min \{ 1,\ \min_j \left(-\frac{y_j^l}{p_j^l} \right) \mid p_j^l < 0 \}$

If $\alpha^l < 1, \quad I = I \cup \{ j^* \}$, where $j^* \in I$ is the index of the minimum value determining α^l.

$l = l + 1$ and return to step 1.

Step 1-3 (Optimality test)

Let $g_q = \min_{j \in I} \{ g_j = \nabla f_j^k + A_j^{\mathrm{T}} \lambda^l \}$ where λ^l is the vector of dual variables associated with constraints $Ap^l = 0$.

If $g_q \geq 0$, STOP. y^l is optimal;

otherwise $I = I - \{ q \}$ and return to step 1.

In the statement of the algorithm we have not indexed the sets I and J with iteration number k, since these sets may change before a step size may be computed in step 4. However the direction of descent which allows an improvement in the objective function f is given index k.

If $f()$ is a strongly convex quadratic function, NAC is a special instance of the classical active set algorithm for solving such problems (see Luenberger, 1984, p. 426). The only difference is that the initial active set is determined by using the current value of the gradient of the Lagrangian. This difference has no consequence on the convergence of the algorithm since, at each iteration l, steps 1 to 3 produce a solution y^{l+1} which is the optimal solution subject to the constraints of the current active set. As the sequence $\{ f(y^l) \}$ is strictly decreasing and since the number of active sets is finite, convergence is achieved in a finite number of iterations.

Theorem 1: The Newton algorithm generates a sequence $\{ x^k \}$, given any initial feasible solution x^1, which converges to the optimal solution of the problem (4)–(6) where f is supposed strongly convex and attains its minimum on Φ.

Proof: Straightforward.

It is interesting to note that in the subproblem of step 1-1, p_I^l may be replaced in the objective function and first constraint, which leads to the subproblem:

$$\min \ \frac{1}{2} p_J^l D_J p_J^l + \nabla f_J^k p_J^l$$

$$\text{subject to } A_J p_J^l = 0$$

The solution of this problem is obtained by solving the system of equations

$$D_J p_J + A_J^T \lambda = -\nabla f_J$$

$$A_J p_J \qquad\quad = 0$$

which is of dimension $(n_J + m_J) \times (n_J + m_J)$, where n_J is the number of variables in the current set J and m_J is the rank of the current matrix A_J, is solved in the following way. The first equation is multiplied by $A_J D_J^{-1}$ to obtain

$$A_J D_J^{-1} D_J p_J^l + A_J D_J^{-1} A_J^T \lambda^l = -A_J D_J^{-1} \nabla f_J$$

which is equivalent to

$$A_J p_J^l + A_J D_J^{-1} A_J^T \lambda^l = -A_J D_J^{-1} \nabla f_J$$

and since $A_J p_J^l = 0$, λ^l is obtained by the equation

$$\lambda^l = -(A_J D_J^{-1} A_J^T)^{-1} A_J D_J^{-1} \nabla f_J$$

and p_J^l is obtained by the equation

$$p_J^l = -D_J^{-1}(\nabla f_J + A_J^T \lambda^l).$$

The system of dimension $(n_J + m_J) \times (n_J + m_J)$ is thus subdivided into two independent systems of sizes $m_J \times m_J$ and $n_J \times n_J$. As D_J is diagonal, so is D_J^{-1} and the computation of p_J^l becomes trivial, reducing the large system to a single system of size $m_J \times m_J$.

In the next section we present a variant of the algorithm.

4. A variant of the Algorithm

In the Newton algorithm presented in Section 1, the quadratic subproblems are solved to ϵ-optimality by the NAC algorithm of the previous section. A variant that

we considered is to stop the NAC algorithm as soon as a feasible descent direction is found. This direction is used to carry out a line search on the objective function of the problem, that is step 2 of the algorithm. The computational efficiency of the original and modified method are similar.

It is worthwhile to note that the NAC algorithm may be easily modified for upper bound constraints of the type $x \leq u$. Without giving all the details of the adaptation, here are the two necessary modifications:

i) In the choice of the initial set of active constraints one adds the additional set of variables in the set

$$I_u = \{ j \in N \mid x_j^k = u_j,\ g_j^k = (A_j^{\mathrm{T}} \lambda^{k-1} + \nabla f_j(x^k)) < 0 \}.$$

ii) The direction of descent must satisfy the inequality constraints. In addition to the constraints $x_j^k \geq 0,\ j \in J$, it is necessary to test the constraints $x_j^k \leq u_i,\ j \in J$.

From a computational viewpoint the upper bound constraints, $x \leq u$, are advantageous since the cardinality of J is likely to be smaller when the set I_u is non empty and, as a consequence, the linear system to solve in order to obtain p_J is smaller as well.

We conclude this section with some general remarks on the NAC algorithm. In the neighbourhood of the optimal solution, the subproblem Q_k is likely to be solved by the first feasible direction found and hence, the variant that we propose on the NAC algorithm will behave as a pure Newton method. Thus, one may expect the locally quadratic rate of convergence of NAC to hold as well for the variant that we propose.

5. Adaptation of the NAC Algorithm for the GSPEP

We proceed to adapt the NAC algorithm for the problem (1)–(3) which is the GSPEP (General Spatial Price Equilibrium Problem). Before proceeding with the adaptation we modify its formulation to that of a single commodity network equilibrium problem (see Dafermos (1980), Drissi (1987), Drissi and Florian (1991)) by augmenting the original network with an additional node m, arcs (m, l) for the origins and arcs (l, m) for the destinations, where

$$\left\{ \begin{array}{ll} d_l = v_{ml}; & S_l(d_l) = C_{ml}(v_{ml}) \\ \text{and } b_l = v_{lm}; & I_l(b_l) = -C_{lm}(v_{lm}). \end{array} \right\} \tag{21}$$

Let A^+ be the set of arcs in the augmented network. Then (1)–(3) may be formulated as

$$\text{Min} \sum_{a \in A^+} \int_0^{v_a} C_a(x)dx \tag{22}$$

$$\text{subject to} \quad \sum_{a \in A_l} v_a - \sum_{a \in B_l} v_a = 0, \quad \forall l \tag{23}$$

$$\text{and} \quad v_a \geq 0, \quad \forall a, \tag{24}$$

where A_l, B_l are defined as before, in the augmented network A^+.

When the right hand side of equation (23) is nonzero and corresponds to demands from origins to destinations, (22)–(24) is a single commodity network equilibrium problem. The adaptation of the NAC algorithm to solve this problem does not require that the right hand side of (23) to be zero and the computational approach is equally applicable to both problems. We suppose that the functions $C_a(.)$ are differentiable and strongly monotone, hence the objective (22) is a strongly convex and twice differentiable function.

Problem (22)–(24) is a particular case of (4)–(6), with the exception that one of the equations in (23) is redundant. In order to ensure that (23) is of full rank it suffices to remove one of the flow conservation equations, say one for node m.

In the following we give the adaptation of the algorithm presented in the previous section for solving (22)–(24). The only difference required to solve a single commodity network flows problem is in the determination of an initial solution.

Step 0 (Initialization)

For the GSPEP $v^1 = 0$ is a feasible solution (for the single commodity network equilibrium problem, an "all-or-nothing" assignment of the demand from the origin to the destination on a shortest path is a feasible solution).

$\lambda^0 = 0$ is a possible initial choice (another choice consists in using the equations of step 2 with $I = \{ j \mid v_j^1 = 0,\ \nabla f(v^1) + A^{\mathrm{T}}\lambda^0 > 0 \})$; $k = 1$.

Step 1-0 (Define the set of initial active constraints)

$g_a^k = (C_a(v_a^k) - \lambda_i^{k-1} - \lambda_j^{k-1})$, $\forall a = (i, j) \in A^+$ is the gradient of the Lagrangian, and $I = \{ a \in A^+ \mid v_a^k = 0 \text{ and } g_a^k > 0 \}$, $J = A^+ - I$.

Step 1-1 (Compute the Newton direction restricted to v_a, $a \in J$)

Since only arcs a, $a \in J$ are considered, the subnetwork used at the current iteration is composed only of these arcs and nodes l that satisfy: $a \in J \mid a = (i, l)$ or $a = (l, j)$.

The following procedure identifies the nodes (constraints) that must be removed in order to ensure that the submatrix A_J is of full rank:

i) $\eta(l) = 0$, $\forall l \in J$.

ii) For each $a = (i, j) \in J$, do $\eta(i) = 1$, $\eta(j) = 1$.

All nodes with $\eta(i) = 0$ are not considered.

λ is the solution of $(A_J D_J^{-1} A_J^{\mathrm{T}})\lambda = -A_J D_J^{-1} \nabla f_J^k$, which is a system of n_J equations and n_J variables. There are two ways to solve this system: one is a pivoting method, such as the Gauss method (see Conte and De Boor (1980)) and the other is an approximate solution by using the conjugate gradient method (Bertsekas, (1983)). The latter method has the advantage that the matrix $(A_J D_J^{-1} A_J^{\mathrm{T}})$ need

not be stored explicitly, which is advantageous for larger problems. However, since the solution is approximate, care must be taken to maintain the solution feasible by using equations (9)–(10) instead of (11)–(12) after every a chosen number of iterations.

a) Computation of $-A_J D_J^{-1} \nabla f_J^k$.

$D_J = \text{diag}(C'_a)$, where $C'_a = \frac{\partial C_a(v_a^k)}{\partial v_a}$. Let $q = -A_J D_J^{-1} \nabla f_J^k$; then

$$q_i = \sum_{a \in A_i \cap J} \frac{C_a}{C'_a} - \sum_{a \in B_i \cap J} \frac{C_a}{C'_a}$$

b) Solution of the system of equations for λ by the pivoting method requires the matrix $M = (A_J D_J^{-1} A_J^{\mathrm{T}})$ which is easily computed by the following procedure:

i) $M := 0$

ii) For each $a = (i, j) \in J$ do $M_{ii} := M_{ii} + 1/C'_a$

$M_{jj} := M_{jj} + 1/C'_a$

$M_{ij} := M_{ij} - 1/C'_a$

$M_{ji} := M_{ji} - 1/C'_a$

The element (i, j) of the matrix $(A_J D_J^{-1} A_J^{\mathrm{T}})$ equals to $\sum_{a \in J} A_J(i, a) A_J(j, a)/C'_a$. However for each a the product $A_J(i, a) A_J(j, a) \neq 0$ if and only if $A_J(i, a) \neq 0$ and $A_J(j, a) \neq 0$. This only occurs when:

- $i = j$ and $a = (i, k)$ or $a = (k, i)$; then $A_J(i, a) A_J(j, a) = (\pm 1)^2 = 1$.
- $i \neq j$ and $a = (i, j)$ or $a = (j, i)$; then $A_J(i, a) A_J(j, a) = \epsilon(-\epsilon)^{-1}$, $\epsilon = \pm 1$.

The matrix M computed by the procedure given above is $(A_J D_J^{-1} A_J^{\mathrm{T}})$. The pivoting algorithm used to solve for λ is given in Appendix I.

c) One may solve for λ by using the conjugate gradient algorithm to solve systems of the types $M\lambda = z$. This method is given in Appendix II. It only requires the explicit knowledge of vectors of the type $r = My$, where y is known.

For our problem, we require the computation of $r = (A_J D_J^{-1} A_J^{\mathrm{T}})y$, without the explicit knowledge of the matrix $(A_J D_J^{-1} A_J^{\mathrm{T}})$. By an argumentation similar to the

above, it is relatively straightforward to show that the vector r may be computed by the procedure

i) $r := 0$

ii) For each $a = (i, j) \in J$ do

$r_i := r_i + (y_i - y_j)/C'_a$

$r_j := r_j + (y_j - y_i)/C'_a$

Computation of $-d_J = -(D_J)^{-1}(\nabla f_J + A_J^{\mathrm{T}}\lambda)$ is acomplished by

$$d_a = \frac{-1}{C'_a}(C_a - \lambda_i + \lambda_j), \quad \forall a = (i, j) \in J.$$

Step 1-2 (Find maximal step length)

Step 1-3 (Remove binding constraints from active set of solution if not optimal)
$g_I = \nabla f_I^k + A_I^{\mathrm{T}}\lambda$ is computed as in step 1 by

$$g_a = C_a + \lambda_j - \lambda_i, \quad \forall a = (i, j) \in I.$$

In the next section, we present computational results obtained with the algorithm.

6. Numerical Results

In the previous section, we adapted the NAC algorithm to solve the GSPEP, and we proposed two approaches to solve the Newton equations (NE). In the following, we call NACG the Newton-Active-Constraints algorithm where the NE are solved by the Gauss method, and we denote NACCG the same algorithm where the NE are solved by a conjugate gradient method. NACG and NACCG have been implemented in a FORTRAN code and the tests were carried out on the CYBER 835 of the Université de Montréal. The termination criterion that has been used is the norm of the reduced gradient, ∇_R: when it is less than a certain tolerance

ϵ: $\|\nabla_R\|_\infty = \max_{j \in J} \{ g_j \} < \epsilon = 10^{-4}$ the computations are stopped. Yet, we would like to note that in the numerical results obtained, the value of the norm of the reduced gradient is of the order of 10^{-6} to 10^{-12}; this is due to the observed quadratic rate of convergence of the method.

We present next the numerical results that demonstrate the efficiency of the proposed algorithms. In order to carry out the tests we generated seven networks, that will be described later, and compared the numerical results obtained by NACG and NACCG on these problems with those obtained by MINOS (Murtagh and Saunders (1977)), which is well known for its efficiency. We note that Friesz and Tobin (1983) proposed a reduced gradient algorithm to solve the GSPEP and this method is the one used in MINOS. Thus, MINOS represents, to our knowledge, the only method that was used so far to solve the GSPEP formulated in terms of arc flow variables.

The seven test networks have been randomly generated, and their dimensions are given in the first five columns of Table 1. The number of variables of the problems is the number of arcs plus the number of destinations; thus, we used problems having between 42 and 691 variables. As mentioned before, the size of the system of Newton equations (NE) to be solved for a given problem is defined by the number of equality constraints, that is, the number of nodes; thus, the size of NE varies between 10×10 and 100×100.

The supply, demand and transportation cost functions were generated as follows:

<u>at origins i:</u> $C_{mi}(v_{mi}) = S_i(v_i) = \alpha_i + \beta_i d_i + \gamma_i d_i^2$

where $\alpha_i \in [0.1, 1]$, $\beta_i \in [0.1, 1]$ and $\gamma_i \in [0.01, 0.05]$ were randomly generated from a uniform distribution.

at destinations j: $-C_{jm}(v_{jm}) = I_j(b_j) = \alpha_j/(1+b_j)$

where $\alpha_j \in [20, 50]$ were randomly generated.

on the arcs a: $C_a(v_a) = \alpha_a + \beta_a v_a + \gamma_a v_a^2$

where $\alpha_a \in [0.1, 1]$, $\beta_a \in [0.1, 1]$ and $\gamma_a \in [0.5, 2]$ were randomly generated.

We note that the transportation cost functions on the arcs are quadratic, thus these problems are denoted by "Q" in Table 1. For problems 1 to 4, we also generated the following types of transportation cost functions:

- linear: $C_a(v_a) = \alpha_a + \beta_a v_a$ that are denoted by "L" in table 1,
- constant: $C_a(v_a) = \alpha_a$ that are denoted by "C" in table 1,

where the α_a and β_a are randomly generated as above.

Before interpreting the numerical results given in Table 1, we would like to insist on the significance of the functions used. In the objective to be minimized, the cost functions associated to the origins are cubic, however, since most of the variables are arc flow variables (32/42 for network 1...), it is clear that the transportation cost functions of type (Q, L, or C) on the arcs have a significant effect on the objective function.

We consider first the problems of type "C". In the objective function, the term $\sum_a \int C_a(x)dx$ is linear and its quadratic approximation is equal to its linear approximation. Thus, the Newton method reduces to a linear method, and the lines "C" in Table 1 reflect this fact for NACG and NACCG. In this case, the quadratic rate of convergence may not be achieved.

When we consider the problems tabulated "L", in the objective function, the term $\sum_a \int C_a(x)dx$ is quadratic, hence it is equal to its quadratic approximation. Therefore, we can expect that the Newton method will give the best results for this kind of problem. Lines "L" in Table 1 show that this is indeed verified for NACG

and NACCG.

In general, for nonlinear problems (L and Q) we see that NACG and NACCG have a better performance than MINOS in terms of the number of iterations (because of the quadratic rate of convergence) as well as in terms of computation times. On the other hand, we see that NACCG is more efficient (computation time) than NCAG, particularly for the larger networks. This is due to the fact that the conjugate gradient method for solving systems of linear equations $M\lambda = z$ does not need to store the matrix M, but only products of the form $r = My$.

Based on these results, we conclude that the NAC method is very efficient method to solve nonlinear GSPE problems, particularly when the systems of linear equations are solved by the conjugate gradient method. Since it does not require the storage of large matrices, NACCG is particularly useful for larger networks; as shown in network 7, NACCG is approximately 6 times better than NAC and 22 times better than MINOS. Although a Gauss method may be used to efficiently solve systems $M\lambda = z$, NACCG appears to be the most efficient algorithm for small networks too.

# Network	# Nodes	# O×D	# Arcs	# Var	Type	NACG			NACCG	MINOS	
						Niter	$\alpha = 0$	s.CPU	s.CPU	Miter	s.CPU
1	10	5 × 5	32	42	C	143	4	3.75	2.68	46	2.46
					L	8	4	0.48	0.55	55	2.97
					Q	9	6	0.51	0.50	51	2.48
2	20	10 × 10	117	137	C	118	7	9.75	14.29	94	8.72
					L	10	6	1.90	1.60	182	14.60
					Q	12	11	2.44	2.43	232	17.60
3	30	15 × 15	251	281	C	489	1	77.00	54.38	137	19.20
					L	10	7	4.53	3.36	369	48.30
					Q	17	12	7.09	5.75	434	54.30
4	40	20 × 20	440	480	C	219	11	68.80	128.34	340	57.10
					L	10	8	9.31	5.91	661	128.00
					Q	11	7	11.80	9.07	918	169.00
5	50	25 × 25	287	337	Q	18	23	20.40	10.03	714	105.00
6	50	25 × 25	549	599	Q	14	24	31.70	14.46	1144	248.00
7	100	50 × 50	591	691	Q	16	24	126.50	22.38	1632	473.00

Table 1 : Numerical results obtained with NACG, NACCG and MINOS

CONCLUSION

In this paper, we considered the general spatial price equilibrium problem. In order to solve such problems, we developed a Newton type algorithm that is combined with an active constraints strategy to handle the nonnegativety constraints or upper bound constraints on the variables. At each iteration of this algorithm, the choice of the initial active set is based on the value of the gradient and of the dual variables. In the neighbourhood of the optimal solution, this choice gives the optimal active set; hence NAC has a locally quadratic (Newton) convergence rate. From a computational viewpoint, we show that the active constraints $x = 0$ (and $x = u$, if any), contribute to reduce the size of the systems of linear equations to be solved at each iteration. We propose two ways of solving the system of linear equations: the Gauss method and the conjugate gradient method.

The two versions of NAC (Gauss and Conjugate gradient to solve the system of linear equations) have been implemented on the CYBER 835 of the Université de Montréal. Several networks have been randomly generated to test the efficiency of the proposed algorithms. For comparison, we used the program MINOS (Murtagh and Saunders (1977)) which is known for its efficiency.

The numerical results obtained demonstrate the most efficient variant of the Newton Active Constraints algorithm is the one that employs the conjugate gradient (CG) method. Since, for the solution of a system of linear equations of type $M\lambda = z$, the CG algorithm does not require the storage of the matrix M, NACCG may be used for large networks. The numerical results show that NACCG is particularly efficient for larger problems; for instance, for a network with 100 nodes (SLE of size 100×100) and 691 variables, NACCG is approximately 22 times faster than MINOS.

We conclude that, for the problems that we tested, NACCG is the best algo-

rithm tried so far to solve the GSPEP; it may be also an attractive algorithm for solving general linearly constrained convex cost minimization problems subject to nonnegativity (and upper bound) constraints on the variables. As directions for further research we identify the possibility of developing other variants of the Newton method combined with an active constraints strategy. The literature contains many variants of the implementation of active constraints methods (see Fletcher, 1972 and Murray and Wright, 1982). We believe that the merit of this paper is to show that one of such approaches is indeed computationally efficient, which does not preclude the development of other variants, which may be efficient as well.

APPENDIX

Solution of $Ax = b$ by a Gauss Method

$A : n \times n,\ x : n \times 1,\ b : n \times 1$

0 - Initialize iord(i) = i, itest = invertible

1 - For $i = 1$ to n, do :

i) $DIV = A_{ii}$

ii) If $DIV = 0$, then: If ($A_{ik} = 0$ for $k = 1$ to n) then

itest = noninvertible

stop

otherwise permute columns i and k where $A_{ik} \neq 0$

permute iord(i) and iord(k)

iii) For $j = 1$ to n, do $A_{ij} = A_{ij}/DIV$

iv) b_i/DIV

v) For $j = 1$ to n, do :

If $j \neq i$ and $A_{ji} \neq 0$, then Ratio = A_{ji}

For $k = i$ to n, do : $A_{jk} = A_{jk} - \text{Ratio} * A_{ik}$

$b_j = b_j - \text{Ratio} * b_i$

2 - For i to n, do : $x(\text{iord}(i)) = b(i)$

Solution of $Hz = -g$ by the Conjugate Gradient Method

$$H : n \times n,\ z : n \times 1,\ g : n \times 1$$

The following algorithm is taken from Bertsekas (1983), where a sequence $\{ z_m \}$ is generated:

$$z_0 = 0,\ z_{m+1} = z_m + \gamma_m p_m, \quad m = 0, 1, \ldots$$

where p_m is obtained by : $p_0 = -Sr_0,\ p_m = -Sr_m + \beta_m p_{m-1}, \quad m = 1, 2, \ldots$

The residual r_m is defined by : $r_m = Hz_m + g, \quad m = 1, 2, \ldots$

and the scalars γ_m and β_m by : $\gamma_m = (r_m^{\mathrm{T}} S r_m)/(p_m^{\mathrm{T}} H p_m), \quad m = 0, 1, \ldots$

$$\beta_m = (r_m^{\mathrm{T}} S r_m)/(r_{m-1}^{\mathrm{T}} S r_{m-1}), \quad m = 1, 2, \ldots$$

In this algorithm, the matrix S is a given definite positive matrix that one can take to be identity matrix, but a choice such $s = \mathrm{diag}(H_{ii})$ may accelerate convergence (Bertsekas, 1982).

The most important aspect to note about this algorithm, is that the matrix H need not be stored explicitly; one needs only products of the form $y = Hv (Hz_m$ and $Hp_m) : n \times 1$.

On the other hand, Bertsekas (1982) notes that only Hp_m has to be computed using H; Hz_m is obtained recursively by $Hz_{m+1} = Hz_m + \gamma_m Hp_m$ because $z_{m+1} = z_m + \gamma_m p_m$; and $Hz_0 = 0$ because $z_0 = 0$.

References

Bertsekas, D.P., "Projected Newton methods for optimization problems with simple constraints". *SIAM Journal of Control and Optimization* 20(2), (1982), pp. 221–246.

Bertsekas, D.P. and Gafni, E.M., "Projected Newton methods and optimization of multicommodity flows". *IEEE Transactions on Automatic Control* AC-28(12), (1983), pp. 1090–1096.

Conte, S.D. and De Boor, C., *Element Numerical Analysis, an Algorithmic Approach*, 3rd edition, Chap. 4, McGraw-Hill, (1980).

Dafermos, S.C., "Traffic equilibrium and variational inequalities". *Transportation Science* 14(1), (1980), pp. 42–54.

Dafermos, S.C. and Nagurney, A., "Isomorphism between spatial price and traffic network equilibrium models". Report LCDS #85-17, Lefschetz Center for Dynamical Systems, Brown University, Providence, RI., (1985).

Drissi-Kaïtouni, O., "Problèmes d'équilibre spatial sur réseau général et méthodes de Newton". *Thèse de Ph.D.*, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, (1987).

Drissi-Kaïtouni, O. and Florian, M., "An algorithm for the spatial price equilibrium problem in term of path flow variables". *Journal of Regional Science* 31(2), (1991), pp. 171–190.

Fletcher, R., "Minimizing general functions subject to linear constraints". In: *Numerical methods for nonlinear otimization*, F.A. Lootsma (Ed.), Academic Press, (1972), pp. 279–296.

Friesz, T.L. and Tobin, R.L., "Formulating and solving the spatial price equilibrium problem with transshipment ir terms of arc variables". *Journal of Regional Science* 23(2), (1983), pp. 187–198.

Luenberger, D.G., *Linear and Nonlinear Programming*, Chap. 6, 7, 11, Addison-Wesley, (1984).

Murtagh, B.A. and Saunders, M.A., *MINOS User's Guide*, Technical Report 77-9, Systems Optimization Laboratory, Stanford University, Stanford, CA, (1977).

Murray, W. and Wright, M.H., "Computation of the search direction in constrained optimization algorithms ". *Mathematical Programming Study* 16, (1982), pp. 62–83.

Ortega, J.M. and Rheinboldt, W.C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, (1970).